LESSON 01: Understanding Core Database Concepts

You are a newly hired junior accountant at a prestigious accounting firm. You have just been tasked with compiling a financial outlook for one of the company's largest clients, with a turnaround time of three weeks. One of the firm's partners feels that the firm isn't getting all the financial data it needs using its existing methods of information retrieval. Presently, the firm uses Excel spreadsheets to create financial outlooks and current financial positions for each of its clients. Upon receiving a total of 15 spreadsheets for the client in question, you quickly realize that without a better way of gathering the information you need, you will not be able to complete your project in the time allotted.

Understanding Database Concepts

THE BOTTOM LINE

Before you begin creating tables and other database elements, you must first understand what databases are and what valuable tools they can be. The benefits of choosing the right type of database will become apparent once the database is used in your company.

A ***database (db)*** is an organized collection of data, typically stored in electronic format. It allows you to input, organize, and retrieve data quickly. Traditional databases are organized by fields, records, and files.

To better understand what a database is, consider the telephone book as a simple example. If you had the telephone book stored on disk, the book would be the file. Within the telephone book, you would have a list of records—each of which contains a name, address, and telephone number. These single pieces of information (name, address, phone number) would each constitute a separate field.

Because a database can store thousands of records, it would be a chore if you had to open a table and go through each record one at a time until you found the record you needed. Of course, the process would be even more difficult if you had to retrieve multiple records.

Thankfully, you don't have to go through database records in this way. Rather, to retrieve data within a database, you run a database ***query***, which is an inquiry into the database that returns information back from the database. In other words, a query is used to ask for information from a database.

If a database contains thousands of records with many fields per record, it may take even a fast computer a significant amount of time to search through a table and retrieve the requested data. This is where a database index comes in handy. An ***index*** is a data structure that improves the speed of data retrieval operations on a database table. The disadvantage of indexes is that they need to be created and updated, which requires processing resources and takes up disk space.

Databases are often found on ***database servers*** so that they can be accessed by multiple users and provide a high level of performance. One popular database server is Microsoft SQL Server. Database servers like SQL Server do not actually house graphical programs, word-processing applications, or any other type of applications. Instead, these servers are entirely optimized to serve only the purposes of the database itself, usually using advanced hardware that can handle the high processing needs of the database. It is also important to note that these servers do not act as workstations; they generally are mounted on racks located in a central data center and can be accessed only through an administrator's desktop system.

Microsoft SQL Server uses three types of files to store databases. Primary data files, which have an .mdf extension, are the first files created in a database and can contain user-defined objects, such as tables and views, as well as system tables that SQL Server requires for keeping track of the database. If the database becomes too large and you run out of room on your first hard disk, you can create secondary data files, which have an .ndf extension, on separate physical hard disks. The third type of file used in SQL Server is a transaction log file. Transaction log files use an .ldf extension and don't contain objects such as tables or views.

Most users do not access a database directly. Instead, they use a ***database management system (DBMS)*** to access it indirectly. A DBMS is a collection of programs that enables you to enter, organize, and select data in a database. For example, a ticket agent may run a ticket system program on his or her desk computer that in turn accesses a ticketing database. There are three types of databases with which you should be familiar in order to make the appropriate choice when developing your own database tables:

- Flat-type databases
- Hierarchical databases
- Relational databases

Each database type has its own important design features.

Understanding Flat-Type Databases

Flat-type databases are simplistic in design. They are most commonly used in plain-text formats. Because their purpose is to hold one record per line, they make access, performance, and queries very quick. An example of this type of database would be what you would find in a .txt or .ini file.

Flat-type databases are considered "flat" because they are two-dimensional ***tables*** consisting of rows and columns. Each column can be referred to as a field (such as a person's last name or a product's ID number), and each row can be referred to as a record (such as a person's or product's information). The following is an example of a simple flat-type database in which a supply company has matched each customer with what he or she consistently orders for easy retrieval and reordering purposes:

id	customer	order
1	allen	notebook
2	smith	paper
3	dennis	pens
4	alex	ink cartridges
5	sloan	printer

CERTIFICATION READY
How are tables organized within a database?
1.1

Understanding Hierarchical Databases

A ***hierarchical database*** is similar to a tree structure (such as a family tree). In this database, each "parent" table can have multiple "children," but each child can have only one parent.

An example of a parent/child hierarchical database is shown in Table 1-1. This database applies to a department of four employees for which the company has just purchased new equipment. Note that one table holds the employee information, whereas the other table holds data about the newly purchased equipment. Here, the table at the top is the "parent," and the table on the bottom is the "child." If several such tables are linked together, the database's tables will start to form a tree structure in which each parent may have multiple child tables and each child table may in turn have children of its own, yet no single child table will have more than one parent.

Parent Table

EMPNUM	FIRSTNAME	LASTNAME	DEPTNUM
100	Paul	Baker	101
101	Jane	Smith	101
102	Jim	Tate	101
103	Ed	Rosen	102

Child Table

SERIALNUM	TYPE	EMPNUM
30032334	Computer	100
4323452	Laptop	101
342342	Monitor	100
234322	Printer	100

Table 1-1 Hierarchical database showing parent and child tables

In this example, the parent table holds the employee data. Each row or record provides an employee's information, including his or her employee number (EmpNum).The child table holds the computer equipment data, and the EmpNum column links each record to the parent table. It is important to note that each piece of equipment must be entered separately. Because we are using a hierarchical database, we can assign multiple computer devices to each employee.

Understanding Relational Databases

The last yet most important database type is the relational database. A ***relational database*** is similar to a hierarchical database in that data is stored in tables and any new information is automatically added into the table without the need to reorganize the table itself. Unlike in hierarchical databases, however, a table in a relational database can have multiple parents.

CERTIFICATION READY
How do relational databases differ from flat-type databases and hierarchical databases?
1.2

An example of a relational database is shown in Table 1-2. The first parent table shows the salespeople within a company, and the second parent table lists what product models are sold by the company. Meanwhile, the child table lists customers who have purchased models from the company; this child table is linked to the first parent table by the SalesNum and to the second parent table by the Model.

Parent Table 1

SalesNum	FirstName	LastName	DeptNum
100	Paul	Baker	101
101	Jane	Smith	101
102	Jim	Tate	101'
103	Ed	Rosen	102

Parent Table 2

Model	Cost	Color
2200MX	$75000	Red
42CRS	$55000	Gray
4232DR	$60000	Red
2201MX	$80000	Blue

Child Table

FirstName	LastName	IDNum	Model	SalesNum
Pete	Wilson	1001	2200MX	100
Jim	Cline	1002	42CRS	101
Omar	Salize	1003	4232DR	103
Louise	Peterson	1004	2201MX	100

Table 1-2 Relational database showing two parent tables and one child table

Understanding Database Fundamentals

A simple database with one table is similar to a spreadsheet that contains rows and columns. However, unlike a spreadsheet, a database allows you to store thousands of rows of data and then access that information more quickly than by reading a spreadsheet.

A spreadsheet is often the starting point for creating a database. With a spreadsheet, it's easy to create headings and start entering data. Adding, deleting, reordering, and formatting headings is simple, and you can easily sort the data under one or more headings. It's also easy to insert, delete, and filter rows matching one or more patterns under a heading.

Many databases will accumulate thousands of rows of data. Depending on your needs, you may wish to create additional tables to hold some of this information. In a spreadsheet, this would be the same as adding additional worksheets.

Spreadsheets, however, are designed for and limited to only thousands of rows per worksheet. Moreover, when a spreadsheet is opened, the whole file is loaded into the computer's memory—so if enough data is stored, the file may eventually fail to load due to insufficient memory. Suddenly, the benefits of using a spreadsheet start to decline. This is when switching to a database makes the most sense.

This comparison highlights three fundamental characteristics of databases:

- They are designed to store billions of rows of data.
- They are limited to the computer's available hard disk space.
- They are optimized to use all a computer's available memory to improve performance.

COMPARING WORKSHEETS TO DATABASE TABLES

As you probably already know, a spreadsheet can contain several worksheets, each of which stores logically grouped information in a tabular format. A worksheet is comparable to a database table, and the headings within it are comparable to the columns or fields within a database table. In a spreadsheet with multiple worksheets, such as that shown in Figure 1-1, each worksheet can be thought of as a different table that belongs to the same database.

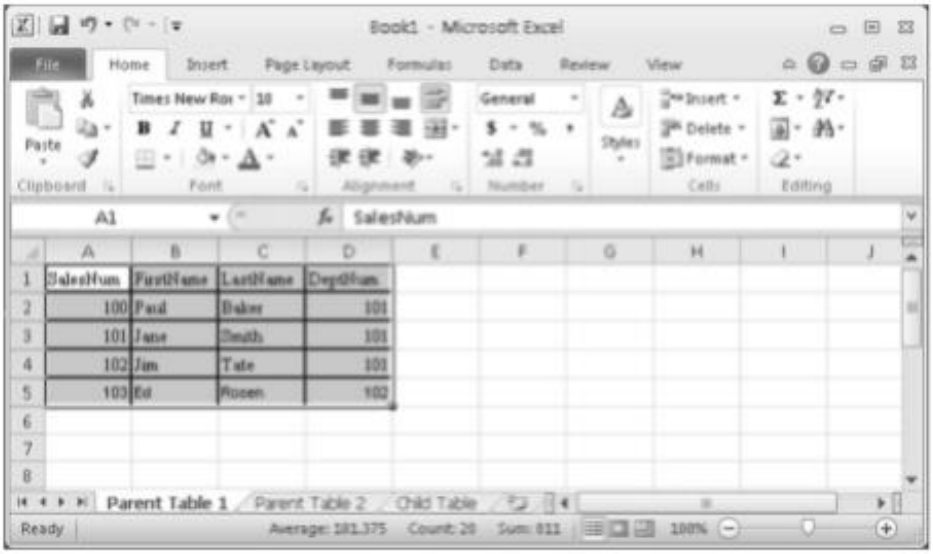

Figure 1-1 Spreadsheet with multiple worksheets

A worksheet column can contain data that might be blank; in such cases, a blank is stored as a Null in the database. A database table can be designed to either allow or disallow Nulls within a column.

UNDERSTANDING CALCULATED VALUES

In a spreadsheet, you can use formulas to calculate values from other information in the same row or column, as shown in Figure 1-2. A calculated value is essentially a value that results from the performance of some sort of calculation or formula on a specified input value. Databases can also be used to generate calculated values, either within the database, within the reports generated from the database, or within the application that is accessing the database.

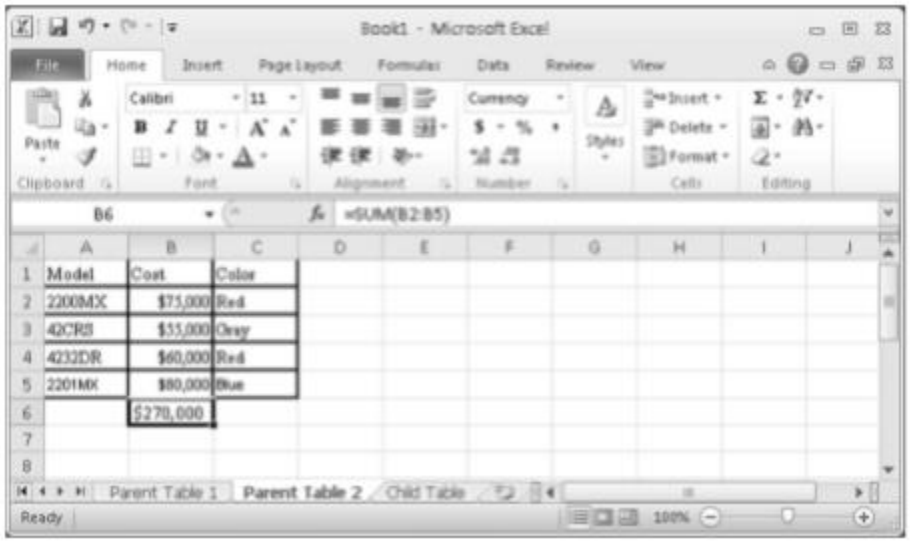

Figure 1-2 Spreadsheet with calculated values

Understanding Relational Database Concepts

Before you design your first relational database, you must understand the elements that form this type of database and the terminology used to describe them.

A relational database helps organize all the data from the various rows and columns of each table, as shown in Figure 1-3. Each column corresponds to one specific type of information you want to store in the database. As you look at the figure, envision each row corresponding to one record, and remember that one instance of each column and each table may be related to one or more other tables.

TAKE NOTE*
To understand relational database models, think about the ways in which a table might relate to one or more other tables.

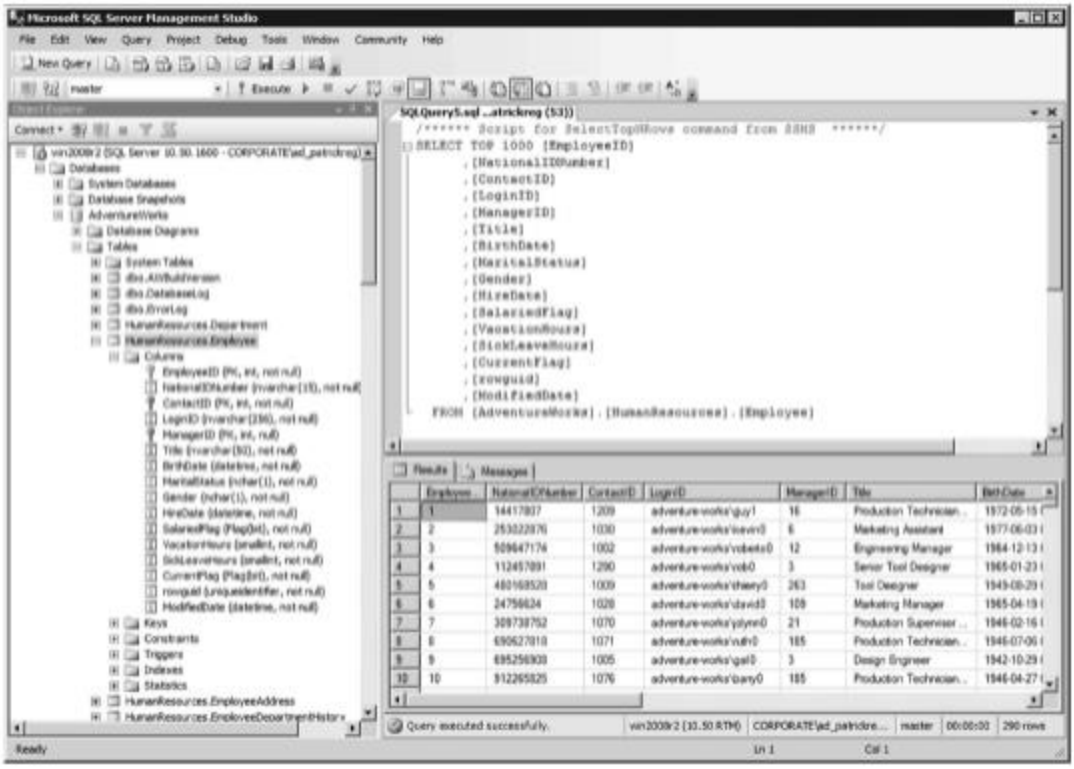

Figure 1-3 Basic database table

A relational database model would organize the data shown in Figure 1-3 into a database table that contains rows and columns, with each column corresponding to one attribute or type of information you want to store. In turn, each row would correspond to one record or one instance of each column.

INTRODUCING LANGUAGE ELEMENTS

Database objects are inherently divided into two broad categories: storage and programmability. A table is structured by columns and rows, and each column then stores data classified as a data type. Figure 1-4 shows sample column attributes for a database. There are a variety of data types to choose from, including built-in types and your own user-defined data types. Data types are discussed in greater detail in Lesson 2.

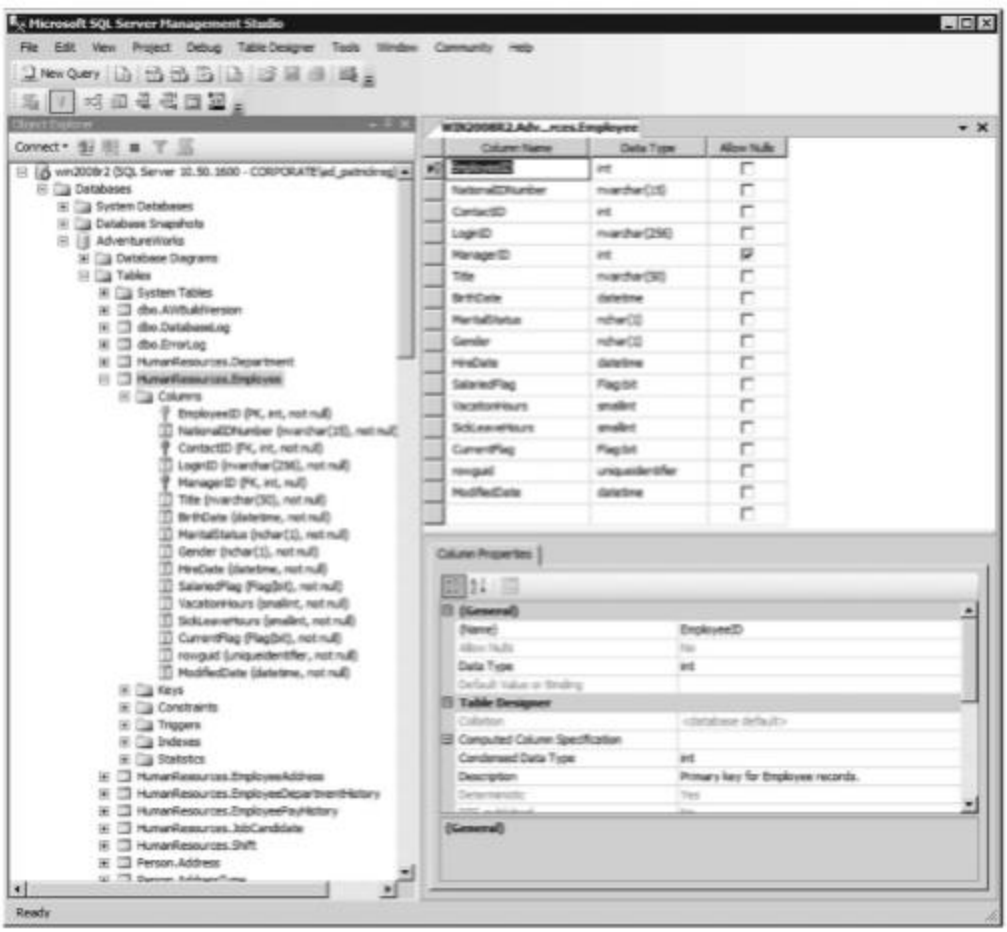

Figure 1-4 Database structure showing column attributes

TAKE NOTE*
Advanced databases, such as SQL Server, periodically analyze queries and create indexes as needed to optimize performance. You can find evidence of this by looking at the index in the database.

Constraints are limitations or rules placed on a field or column to ensure that data that is considered invalid is not entered. For example, if a person's age should be input, then the data that is entered must be a positive number—a person cannot have a negative age. A variety of constraints are available with SQL Server 2008, including the following:

- A unique constraint allows the database administrator to specifically identify which column should not contain duplicate values.
- A check constraint allows the administrator to limit the types of data a user can insert into the database.
- A default constraint is used to insert a default value into a column. If no other value is specified, the default value will be added to all new records.

- A not null constraint ensures that data is entered into a cell. In other words, the cell cannot be blank. It also means that you cannot insert a new record or update a record without adding a value to this field.
- The primary key constraint uniquely identifies each record in a database table. The primary key must contain unique values and it cannot contain NULL values. Each table should have a primary key, and each table can have only *one* primary key.
- A foreign key constraint in one table points to a primary key in another table.

For an example of database constraints, see Figure 1-5.

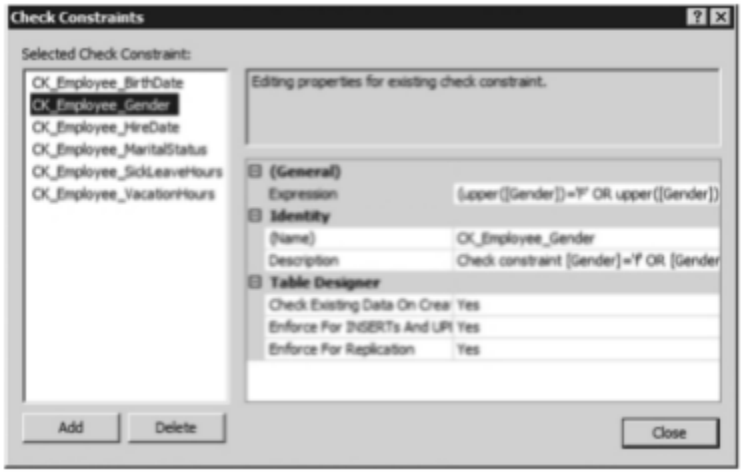

Figure 1-5 A database constraint

TAKE NOTE*
A foreign key is also known as a self-reference.

Columns marked as foreign keys can contain null values. This is *not* a desirable standard of practice, however, because it may be impossible to verify the constraints if a foreign key consists of two or more columns and contains null values. This means the integrity of your data cannot be guaranteed.

It is also possible for a foreign key constraint to reference columns in the same table; this is known as a self-reference. When a self-reference is used to query a table, this arrangement is now referenced as a self-join. As an example of a self-reference table, say you want to create a Generations table that contains names of people using columns named PersonID, PersonName, and MotherID. The mother is also a person stored in the Generations table, so you can create a foreign key relationship from the MotherID (the foreign key column) referencing PersonID (the primary key column).

Using the SQL Server Management Studio Interface

When you install Microsoft SQL Server, you also install the ***SQL Server Management Studio (SSMS)***, which is the primary tool for managing the server and its databases using a graphical interface.

The central feature of SSMS is the Object Explorer, which allows users to browse, select, and manage any of the objects within the server (see Figure 1-6). SSMS can also be used to view and optimize database performance, as well as to create and modify databases, tables, and indexes.

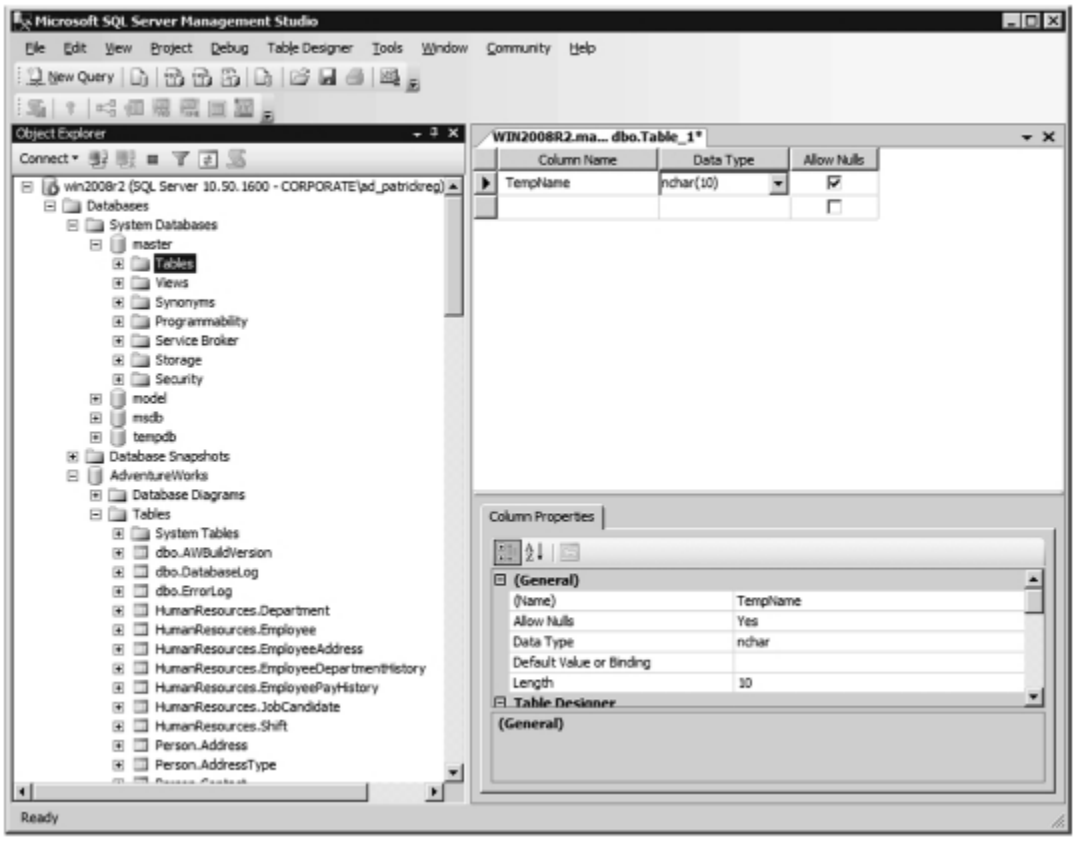

Figure 1-6 SQL Server Management Studio

In addition, SSMS includes the Query Analyzer (see Figure 1-7), which provides a GUI-based interface to write and execute queries. The Query Analyzer supports the following:

- ***XQuery*** is a query and functional programming language that is designed to query collections of XML data.

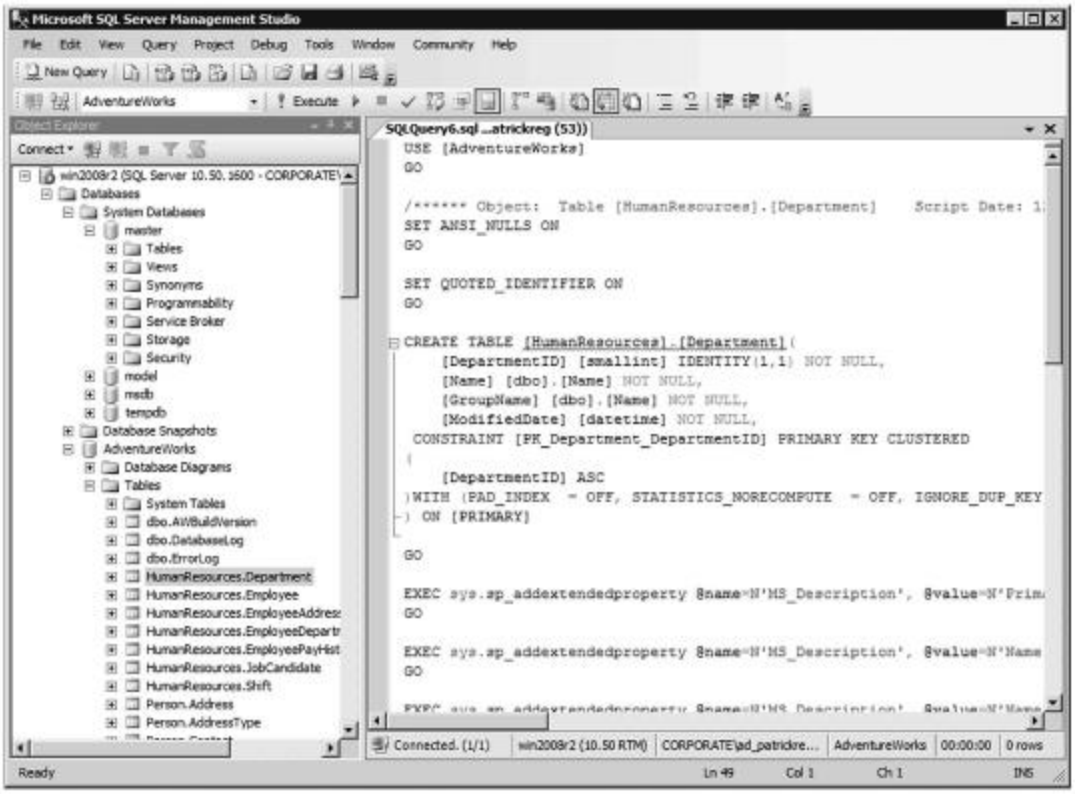

Figure 1-7 Query Analyzer

- ***SQLCMD*** is a command-line application that comes with Microsoft SQL Server and exposes the management features of SQL Server. It allows SQL queries to be written and executed from the command prompt. It can also act as a scripting language to create and run a set of SQL statements as a script. Such scripts are stored as .sql files, and they are used either for management of databases or to create the database schema during database deployment.
- ***Transact-SQL*** is the primary means of programming and managing SQL Server. It exposes keywords so that you can create and manage databases and their components and monitor and manage the server itself. When you use SSMS to perform an action or task, you are executing Transact-SQL commands.

Note that you must have SQL Server 2008 installed on your system before moving on to the next section.

LOAD THE SSMS INTERFACE

GET READY. Before you begin these steps, be sure to launch SSMS.

1. Click the Start button, then click Microsoft SQL Server 2008 to expand the program selection.
2. Click SQL Server Management Studio. The Management Studio opens, displaying the Connect to Server dialog box.
3. Change the server connection details (if necessary) and click Connect. After you have configured the server setting correctly, the SQL Server Management Studio Interface will be visible.

+ TROUBLESHOOTING

Your computer may not have the SSMS interface installed as part of the SQL Server 2008 program. If you cannot find the Management Studio tool when you look under Program Files, you may have to add it as a server installation update. To do this, insert the installation CD and click the Advanced button in the Components to Install window when it appears.

TAKE NOTE*

You can also install SQL Server Management Studio on any Windows desktop operating system so that you can remotely connect to and manage a SQL server.

PAUSE. Leave the SQL Server Management Studio open for the remainder of the lesson.

The SQL Server Management Studio can be used to perform most of the activities you are required to do and can be considered a "one-stop" tool.

CREATE A DATABASE WITH THE SSMS INTERFACE

GET READY. Before you can start managing databases, you must first create them. To do so, follow these steps:

1. Open SSMS by clicking Start > All Programs > Microsoft SQL Server 2008 > SQL Server Management Studio.
2. Make sure the Database Engine is selected, then click the Connect button.
3. Click the plus sign (+) next to Databases to expand it.
4. Right-click Databases, then select New Databases from the menu that appears.
5. In the Database name field, type the name of the database you want to create. Then click the OK button.

DELETE A DATABASE WITH THE SSMS INTERFACE

GET READY. From time to time, you may want to remove databases that are no longer being used. To do so using the SSMS interface, follow these steps:

1. Open SSMS by clicking Start > All Programs > Microsoft SQL Server 2008 > SQL Server Management Studio.
2. Make sure the Database Engine is selected, then click the Connect button.
3. Click the plus sign (+) next to Databases to expand it.
4. Right-click the name of the database you want to delete, then select Delete from the menu that appears.
5. Select Close Existing Connections, then click the OK button.

It's important to note that SQL Server has an extensive help area. In addition, when you install SQL Server, you have the option to install Books Online and Server Tutorials.

Therefore, if you want to find information about a certain option or command, you should start by checking these resources. Of course, if you still cannot find what you are looking for, don't be afraid to search the Internet.

Understanding Data Manipulation Language (DML)

THE BOTTOM LINE

When creating databases, it is important to understand what language elements can do within your database structure.

Data Manipulation Language (DML) is the language element that allows you to use the core statements `INSERT`, `UPDATE`, `DELETE`, and `MERGE` to manipulate data in any SQL Server tables. Core DML statements include the following:

- `SELECT`: Retrieves rows from the database and enables the selection of one or many rows or columns from one or many tables in SQL Server.

  CERTIFICATION READY

  Which popular commands used with SQL are DML commands?

  1.3

- `INSERT`: Adds one or more new rows to a table or a view in SQL Server.
- `UPDATE`: Changes existing data in one or more columns in a table or view.
- `DELETE`: Removes rows from a table or view.
- `MERGE`: Performs insert, update, or delete operations on a target table based on the results of a join with a source table.

When you use DML statements such as INSERT, UPDATE, DELETE, or MERGE, you need to realize that on the whole, they either succeed or fail. For example, if you were to try to insert 10,000 records into a table but violated one of the primary key or unique constraints, the entire 10,000 rows of records would roll back immediately and not one record would be inserted into the table. Similarly, if a DELETE statement violates a foreign key constraint (even in just one row), nothing would be deleted. Therefore, when using DML statements, you must query the table to verify that key constraints are met and your syntax is correct. The following section shows the correct syntax for working with constraints.

Understanding Data Definition Language (DDL)

THE BOTTOM LINE

Data Definition Language (DDL) statements form part of the Transact-SQL portion of SQL Server and can be used to create database objects such as tables and views.

Data Definition Language (DDL) is a subset of the Transact-SQL language; it deals with creating database objects like tables, constraints, and stored procedures. The interface used to create these underlying DDL statements is the SSMS user interface, as shown in Figure 1-8.

CERTIFICATION READY

What popular SQL commands are DDL commands?
1.4

Figure 1-8 SQL Server Management Studio user interface

Using DDL Statements

The SSMS user interface allows you to visually design DDL statements. A DDL script statement task can always be completed through the SSMS user interface, but not all the options you may wish to use with the DDL script can be accomplished through this interface. Therefore, you must be familiar with the DDL statements `USE`, `CREATE`, `ALTER`, and `DROP` in order to create and manage tables, user-defined data types, views, triggers, functions, and stored procedures.

Although most DDL statements can be executed using the SSMS graphical interface, you still have more power, flexibility, and control when using DDL statements themselves. You can also use DDL statements to script tasks or activities that can be scheduled or executed as needed. Again, the six main DDL statements are as follows:

- `USE`: Changes the database context.
- `CREATE`: Creates a SQL Server database object (table, view, or stored procedure).
- `ALTER`: Changes an existing object.
- `DROP`: Removes an object from the database.
- `TRUNCATE`: Removes rows from a table and frees the space used by those rows.
- `DELETE`: Remove rows from a table but does not free the space used by those rows removed.

Let's go through each of these key DDL statements with further explanation and an example of each.

CERTIFICATION READY
What DDL command would you use to change the database content, and which would you use to create a table?
1.4

USE

One Transact-SQL command worth mentioning is the `USE` command. The `USE` command changes the database context to the specified database or database snapshot. In other words, when you perform commands on a particular database, you will likely have to enter the `USE` command to select the database first. For example, to select a database named TESTDB, you would execute the following command:

```
USE TESTDB
```

CREATE

The CREATE statement allows you to create a variety of database objects, including tables, views, and stored procedures. For instance, say you want to create a new table named Planets within a database named AdventureWorks. To do this, you would use the following sequence of commands:

```
USE [AdventureWorks]
GO
CREATE TABLE [dbo].[Planets](
     [IndvidualID] [int] NOT NULL,
     [PlanetName] [varchar](50) NULL,
     [PlanetType] [varchar](50) NULL,
     [Radius] [varchar](50) NULL,
     [TimeCreated] [datetime] NULL
) ON [PRIMARY]
GO
```

Here, the use of [AdventureWorks] changes the database context to AdventureWorks, and the GO command executes the previous set of commands. The CREATE TABLE [dbo].[Planets] command is used to create the Planets table. IndividualID, PlanetName, PlanetType, Radius, and TimeCreated are the columns within the Planets table. Individual ID cannot be NULL. Int, varchar, and datetime specify the data type, which describes what type of data can be entered into the column. (Data types will be explained in detail in Lesson 2.)

ALTER

The ALTER statement changes an existing object; you can use it to add or remove columns from a table, as shown in the following example:

```
ALTER TABLE Shirt
ADD Price Money;
GO
```

Here, ALTER was used to add a Price column to the Shirt table. If you then wanted to set the prices in this column, you could use the UPDATE statement as follows:

```
UPDATE Shirt SET Price = 13.50 WHERE ProductID = 1;
UPDATE Shirt SET Price = 13.50 WHERE ProductID = 2;
UPDATE Shirt SET Price = 10.00 WHERE ProductID = 3;
UPDATE Shirt SET Price = 12.00 WHERE ProductID = 4;
GO
```

You can also use ALTER to change the definition of a view, stored procedure, trigger, or function. For instance, the following command sequence redefines the view to include the Price column:

```
ALTER VIEW Size AS
SELECT ProductID, ProductName, Price FROM Shirt
WHERE ProductType = 'Size';
GO
SELECT * FROM Size
-- Results:
-- ProductID        ProductName      Price
-- -------------    -------------    -------------
-- 1                Red              13.50
-- 2                Blue             13.50
-- 3                Orange           10.00
-- 4                Black            12.00
```

When working with these statements, be careful that you don't confuse ALTER with UPDATE . Remember, ALTER changes the object definition, but UPDATE changes the data in the table.

DROP

The DROP statement actually removes an object from a database, but if other objects are dependent on the object you are attempting to remove, this statement will fail and an error will be raised. The following example shows how you could use DROP to delete data from the Shirts table, then remove it from the Sizes view, and finally remove the

Shirts table from the database. In this example, we also try to drop the Person.Contact table, but, as you will notice, this operation fails because there are other objects that are dependent on the Person. Contact table:

```
DELETE FROM Shirt
Select * FROM Size
-- Results:
     -- ProductID             ProductName       Price
     -- -------------         -------------     -------------
-- (0 row(s) affected)
DROP VIEW Size;
GO
DROP TABLE Person.Contact
-- Results:
-- Msg 3726, Level 16, State 1, Line 1
-- Could not drop object 'Person.Contact' because it is referenced by a
FOREIGN KEY constraint.
```

Remember to not confuse DROP, which removes an object from the database, with DELETE, which deletes data from within a table.

TRUNCATE AND DELETE

Two other DDL statements with which you should be familiar are TRUNCATE and DELETE. The DELETE statement is used to delete rows from a table, but it does not free the space containing the table. In comparison, the SQL TRUNCATE command is used to both delete the rows from a table and free the space containing the table. Thus, to delete all rows from a table named User, you would enter the following command:

```
DELETE FROM user;
```

Similarly, to delete an employee with the identification number 200 from the User table, you would enter the following command:

```
DELETE FROM employee; DELETE FROM user WHERE id = 200;
```

TAKE NOTE*
If you are deleting data from tables in a large database, use TRUNCATE : it is more efficient. Use DELETE for smaller databases.

SYSTEM TABLES

When you want to query system views to verify whether the object(s) you wish to drop are, in fact, in the database tables, you need to know what tables are the most useful. System views belong to the sys schema. Some of these system tables include the following:

- sys.Tables
- sys.Columns
- sys.Databases
- sys.Constraints
- sys.Views
- sys.Procedures
- sys.Indexes
- sys.Triggers
- sys.Objects

All of these view names are self-explanatory. For instance, the sys.Objects view contains a row for every object in the database with key column names of name, object_id, type_desc, type, create_date, and modify_date.

SKILL SUMMARY

IN THIS LESSON, YOU LEARNED THE FOLLOWING:

- A database (db) is an organized collection of data, typically stored in electronic format. It allows you to input, organize, and retrieve data quickly.
- Microsoft SQL Server uses three types of files to store databases. Primary data files, with an .mdf extension, are the first files created in a database and can contain user-defined objects, such as tables and views, as well as system tables required by SQL Server to keep track of the database.
- If a database gets too large and you run out of room on your first hard disk, you can create secondary data files, with an .ndf extension, on separate physical hard disks to give your database more room.
- The third type of file used by SQL Server is a transaction log file. Transaction log files use an .ldf extension and don't contain any objects such as tables or views.
- To retrieve data within a database, you run a database query. In other words, a query is used to ask for information from the database and data is then returned.
- A database index is a data structure that improves the speed of data retrieval operations on a database table.
- Most users do not access databases directly. Instead, they use a database management system (DBMS) to access them indirectly.
- A flat-type database is simplistic in design. These databases are most commonly used in plain-text formats, and their purpose is to hold one record per line, making access and queries very rapid.
- Tables, used to store data, are two-dimensional objects consisting of rows and columns.
- A hierarchical database is similar to a tree structure (such as a family tree). Each parent table can have multiple children, but each child table can have only one parent.

IN THIS LESSON, YOU LEARNED THE FOLLOWING:

- A relational database is similar to a hierarchical database in that data is stored in tables and any new information is automatically added to the table without the need to reorganize the table itself. Unlike tables in a hierarchical database, however, a table in a relational database can have multiple parents.
- Databases are often put on database servers so that they can be accessed by multiple users and provide a higher level of performance. One popular database server is Microsoft SQL Server.
- Constraints are limitations or rules placed on a field or column to ensure that data that is considered invalid is not entered.
- SQL Server Management Studio (SSMS) is the primary tool to manage a server and its databases using a graphical interface.
- Data Manipulation Language (DML) is the language element that allows you to use the core statements `INSERT` , `UPDATE` , `DELETE` , and `MERGE` to manipulate data in any SQL Server tables.
- Data Definition Language (DDL) is a subset of the Transact-SQL language; it deals with creating database objects like tables, constraints, and stored procedures.